УДК 004.94
А.М. Луцків, канд. техн. наук, доцент, М.В. Ващук
(Тернопільський національний технічний університет імені Івана Пулюя)

ГРАМАТИКА ПЕРЕТВОРЕННЯ ПАРАМЕТРІВ МОДЕЛІ МЕРЕЖІ ПЕТРІ У ПРОГРАМНИЙ КОД МОВИ С++

UDC 004.94
A.M. Lutskiv PhD., Assoc. Prof., M.V. Vashschuk

GRAMMAR OF PETRI NETWORK MODEL PARAMETERS TRANSFORMATION IN C ++

Мережі Петрі можуть застосовуватись для того, щоб забезпечити ефективність моделювання паралельних асинхронних процесів, які протікають у динамічних комп'ютерних системах та здійснити алгоритмічний опис об'єкту. Мережі Петрі, з точки зору їх імплементації, поділяють на три основні класи, що відрізняються між собою принципами організації позицій та переходів у мережі.

До цих трьох класів входять:

– безпечні МП – мережі, у вершинах (позиціях) яких при виконанні переходу(ів) може бути відсутня або наявна лише одна мітка;

– обмежені МП – мережі, в яких у вершинах міститься цілочисельне значення міток, а дуги, у вигляді цілих чисел, визначають кількісний їх розподіл у мережі після того, як вони пройшли через переходи (перехід спрацював);

– Е-мережі Петрі – тип мереж, де переходи можуть належати до кількох наперед визначених видів, і спрацювання яких відбувається лише за наявності визначеної кількості міток у мережі, а складні макропереходи здатні до зміни своєї структури.

Для реалізації граматики трансляції мереж Петрі обрано одну з найбільш використовуваних мов програмування – С++, яка є лідером у застосуванні у сфері системного програмування.

Для представлення структури будь-якої програми за допомогою мови високого рівня програмування можна представити у вигляді сукупності блоків:

<заголовок програми>
<ініціалізація вхідних даних>
<оголошення необхідних об'єктів>
<змінні>
<масиви>
<структури>
<функції>
<ініціалізація початковими значеннями>
<блок операторів (тіло програми)>
< оператори завершення роботи програми>

Правила, які визначені граматикою трансляції, повинні передбачати операції заповнення відповідних блоків коду програми.

Початковий символ граматики визначає запис заголовку програми і оператора завершення:

$S \rightarrow$ *void main () { <лексема> }*

<лексема> $\rightarrow$ *<P>|<T>|<A>|*$<M_0>|<G_0>|<t_i>$

<P> $\rightarrow$ *int Np = P.atr*

Ініціалізація змінної *Np* (кількість позицій) та ініціалізація значення атрибуту *P.atr.*

<P> $\rightarrow$ *int masM [P.atr]*

Ініціалізація масиву *masM* для зберігання даних розмітки розмірністю *P.atr.*

<T> $\rightarrow$ *int Nt = T.atr*

Ініціалізація змінної *Nt* (кількість переходів) і присвоєння їй значення атрибута *T.atr.*

<A>→ int Na = A.atr

Ініціалізація змінної *Na* (розмірність вектора комірок пам'яті) і присвоєння їй значення атрибута *A.atr.*

<A>→ int masA [A.atr] [P.atr]

Ініціалізація двомірного масиву *masA* для зберігання значень векторів комірок пам'яті мережі розмірністю *P.atr.*

$<M_0> \rightarrow \{ masM\ [N.atr]=1 \mid <M_0>\}_{N \in P}$

Заповнення початковими значеннями масиву поточної розмітки, значення *masM [i]=1* характеризує наявність мітки в позиції p_i.

$<G_0> \rightarrow \{ masA\ [A.atr]\ [P.atr]= N.atr \mid <G_0>\}_{N \in P,\ ai \in P}$

Заповнення початковими значеннями масиву стану векторів комірок пам'яті у відповідності до значень *N.atr*.

<t_i>→ func_t T.atr () {

<затримка спрацювання переходу>

<перевірка умови>

<переміщення міток>

<перетворення вектора пам'яті> }

Ініціалізація функції *func_t<i>*, що визначає повний набір правил функціонування переходу t_i.

<t_i>→ struct_t T.atr () {

int <X> *// початкові позиції переходу*

int <Y> *// вихідні позиції переходу*

int <TR> *// затримка спрацювання переходу*

struct_R<r_i> () {

int <RI> *// умова спрацювання переходу*

int <PM> *// видалити мітку з*

int <AM> } *// додати мітку в*

struct_RO<ρ_i> () {

int <ROI> // правило перетворення атрибутів мітки

int <C1> // ліва частина правила

int <C2> } } *// права частина правила*

Ініціалізація структури *struct_t<i>*, значення полів якої містять дані повного опису переходу t_i.

<t_i>→ if <умова спрацювання переходу> then func_t T.atr

Заповнення <тіла програми>. Стрічка коду описує виклик на виконання функції *func_t<i>* при істинності умови спрацювання переходу t_i.

Реалізація моделі формального опису в конкретному програмному чи програмно-апаратному середовищі є невід'ємною частиною процесу створення динамічних комп'ютерних систем.